



Praktikum zur Veranstaltung XML-Technologie: **Übung 06**

Das „MOM“-Beispiel mittels
XML Schema, Teil 1



Organisatorisches



- Arbeitsverzeichnis:
`~/kurse/xmltech/06/`
- Dateinamen:
`06-bestell.xml` `XMLSchema.dtd`
`06-bestell.dtd` `datatypes.dtd`
`06-bestell.xsd`
- Abzugeben:
`06-bestell.xml`, `06-bestell.xsd`
- Werkzeuge:
`emacs` `# NICHT X-Emacs`
`SAXPrint` `# bzw. "schemavalidator"`
`mozilla` `# zur Nachkontrolle`



- Zur Aufgabe:
 - Ziel ist der Umbau der DTD zur Bestellung in ein XML Schema.
 - Zweistufiges Vorgehen:
 - Aufgabe 06:
 - Zunächst reiner Umbau DTD → Schema
 - Dabei Kennenlernen der neuen Werkzeuge
 - Aufgabe 07:
 - Präzisere Datentypen u.a. Schema-Features!
- Abgabezeitpunkt:
 - Ausnahmsweise erst zusammen mit Teil 2 (Aufgabe 07).



- Dateien:
 - 06-bestell.xml, 06-bestell.dtd
 - Aus dem Dozentenverzeichnis kopieren
 - Dann geeignet modifizieren!
 - Hinweis:
 - Die beiden Dateien entsprechen den Versionen aus Aufgabe 04, sind aber um das Element „Bestellungen“ erweitert worden, um auch mehrere Bestellungen in einem Dokument erfassen zu können.
 - XMLSchema.dtd, datatypes.dtd
 - Aus dem Dozentenverzeichnis kopieren
 - 06-bestell.xsd
 - Selbständig aufzubauen!



- Werkzeuge:
 - Der Validierer des Emacs ist nicht für Namensräume und XML Schema geeignet.
 - Wir verwenden als Schema-Validierer daher ein neues Werkzeug, das auf der C++ Version der Bibliothek „Xerces“ der Apache Foundation basiert.
 - Xerces ist ein validierender XML-Parser und auch voll *namespace*- und XML Schema-kompatibel!
 - Das Kommandozeilen-Werkzeug „SAXPrint“ schreibt XML-Dateien eigentlich nur „nett formatiert“ nach **stdout**. Implizit prüft der Xerces-Parser die Quelldaten aber und schreibt alle Fehler und Warnungen nach **stderr**.
 - Uns interessiert hier meist nur der Fehler-Output nach **stderr**!



- „schemavalidator“:
 - Legen Sie in ~/.bashrc folgenden neuen alias-Eintrag an, z.B. durch Anfügen folgender Zeile:

```
alias schemavalidator=  
  '/usr/local/xerces-c-2.1.0/bin/SAXPrint -n -s -v=always'
```

- Schließen Sie Ihre aktuelle „bash“ und öffnen Sie eine neue. Nun sollte das neue „Alias“ funktionieren.
- Test:

```
$ schemavalidator 06-bestell.xml > tmpbestell.xml  
$ diff 06-bestell.xml tmpbestell.xml
```

 - Hat Ihr „Validator“-Aufruf funktioniert und Output erzeugt?
 - Wie erklären Sie den „diff“-Output??



Vorbereitungen



- Test-Ergänzung (optional) ohne spezielle Optionen:

```
$ /usr/local/xerces-c-2.1.0/bin/SAXPrint  
    06-bestell.xml > tmpbestell.xml  
$ diff 06-bestell.xml tmpbestell.xml
```

- Wie erklären Sie diesen „diff“-Output?
- Hinweis: Beachten Sie, dass SAXPrint diesmal ohne die Optionen `-n`, `-s`, `-v` aufgerufen wurde.



Aufgabe



- A: Verbindung der XML-Datei mit der Schemadatei
 - Legen Sie eine zunächst leere Schemadatei `06-bestell.xsd` an.
 - Bauen Sie nun in `06-bestell.xml` die Attribute ein, mit deren Hilfe der Validierer die Schema-Datei finden kann.
 - Testen Sie, ob der Validierer die Schema-Datei auch findet:

```
schemavalidator 06-bestell.xml > /dev/null
```

- Erst wenn Sie mindestens eine Fehlermeldung aus der Schemadatei `06-bestell.xsd` erhalten, können Sie sicher sein, dass diese Datei auch gefunden wurde.
Provozieren Sie ggf. einen Fehler, etwa durch Angabe einer syntaktisch falschen Zeile in der Schemadatei!



Aufgabe



- B: Vorbereitung der Schemadatei
 - Tragen Sie in die Schemadatei `06-bestell.xsd` ggf. einen XML-Prolog ein (s. nächste Seite).
 - Legen Sie Namensräume & Präfix-Werte fest, erstellen Sie dann entsprechend das *start tag* und *end tag* für „Schema“ (incl. der gewählten Präfix-Angaben).
 - Ein Beispiel dazu finden Sie in der Vorlesung - passen Sie es an.
 - Testen Sie mittels Schema-Validierung, ob Ihre Erweiterung funktioniert.
 - Sie sollten nur (zahlreiche) Fehler über inhaltliche Probleme wie unbekannte Elemente und nicht deklarierte Attribute erhalten, aber keine über das Dokumenten-Element selbst.



Aufgabe



- B: Vorbereitung der Schemadatei (**optional**)
 - Auch für XML Schema gibt es eine DTD. Sie ist nicht normativ, aber nützlich, etwa für den XML-Modus des Emacs.
 - Sie besteht aus den Dateien `XMLSchema.dtd` und `datatypes.dtd`. Kopieren Sie beide Dateien aus dem Dozentenverzeichnis zu dieser Aufgabe in Ihr Arbeitsverzeichnis „06“.
 - Bauen Sie eine Dokumententyp-Deklaration in den Prolog Ihrer Schema-Datei ein mit Verweis auf „`XMLSchema.dtd`“.
 - Nun können Sie die Schema-Datei mit dem Emacs komfortabler erstellen und auch wie gewohnt (XML-) validieren.
 - Hinweise:
 - Man muss *parameter entities* „p“ und „s“ (Präfix und Suffix) sowie das Attribut „`xmlns`“ noch geeignet im internen *subset* deklarieren...
 - Leider mag „SAXPrint“ diese DTD nicht, wenn er schema-validiert!
 - Ausweg: `XMLSchema.dtd` wechselweise ein- oder ausblenden, je nachdem ob man mit Emacs oder `schemavalidator` arbeitet. ☹



Aufgabe



- C: Übertragung der DTD - Anlegen der Elemente
 - Legen Sie für jedes Element der DTD im Schema ein **e**lement-Element an.
 - Verwenden Sie als Datentyp des Inhalts zunächst nur „string“, wo bisher „#PCDATA“ stand.
 - Verweisen Sie auf enthaltene Elemente mittels Attribut „ref“ – bitte KEINE Elemente implizit anlegen.
 - Prüfen Sie mittels schemavalidator, ob Sie Fortschritte machen (weniger Fehler) oder neue Fehler entstehen.
- Hinweise:
 - Beispiele für die Übertragung von DTD-Komponenten finden Sie im Vorlesungsmaterial.



Aufgabe



- D: Übertragung der DTD – Die Attribute
 - Ergänzen Sie die **e**lement-Elemente nun ggf. um **a**tttribute-Elemente. Bauen Sie wenn nötig die Elemente so um, dass sie Attribute aufnehmen können.
 - Verwenden Sie als Datentypen der Attribute zunächst nur die direkt aus den DTDs stammenden Typen wie „string“ für „CDATA“, „NMTOKEN“, „NOTATION“, „ID“, etc.
 - Verweisen Sie ggf. auf **n**otation-Elemente.
 - Prüfen Sie mittels **s**chemavalidator, ob Sie Fortschritte machen (weniger Fehler) oder neue Fehler entstehen.
 - Es sollten nur noch Fehler bez. vermischer Notationen übrigbleiben.



- E: Übertragung der DTD – Die Notationen
 - Was in der DTD eine Notation war, bleibt auch in Schema eine. Legen Sie daher die entsprechenden **notation**-Elemente an.
 - Prüfen Sie erneut mittels **schemavalidator**. Die Fehler sollten nun verschwinden!
- Hinweise
 - Das Verständnis der Attribute **system** und **public** ist offenbar in Schema nicht direkt an XML's „PublicID“ angelehnt. Verwenden Sie Attribut „**public**“, wenn „**system**“ allein nicht akzeptiert wird!



- Übertragung der DTD in ein XML Schema
 - Wir haben nun die DTD in ein Schema „1:1“ übersetzt. Damit sind die Voraussetzungen geschaffen, um die eigentlichen Vorzüge von XML Schema kennenzulernen!
 - Teil 2 (Aufgabe 07) zu XML Schema besteht i.w. aus dem Definieren eigener, an die Anforderungen aus Aufgabe 04 an Bestelldaten angepasster Datentypen sowie deren Verwendung.
 - Ziel ist dabei die möglichst präzise Datenmodellierung und damit die Erkennung von fehlerhaften Inhalten möglichst schon auf Senderseite.